NAME CONVERTS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; ;

; CONVERTS ;

; Conversion Functions ;

; EE/CS 51 ;

; Archan Luhar ;

; TA: Joe Greef ;

; ;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; file description including table of contents

;

; Revision History:

; 1/26/06 Glen George initial revision

; 10/26/13 Archan Luhar Finished Homework 2

; PREPROCESSOR DEFINITIONS

ASCII\_NULL EQU 0

; START CODE

CGROUP GROUP CODE

CODE SEGMENT PUBLIC 'CODE'

ASSUME CS:CGROUP

; Dec2String

;

; Description: This function is used to create a decimal ascii string

; given a signed binary value.

;

; Operation: Given a 16 bit signed value, the function writes to

; specified memory location the ascii string representing the

; number in base 10 by looping over the number to find

; the ones, tens, etc. digits. The maximum number of digits

; is 5. The maximum number of bytes that can be written, thus,

; are 7 (+1 '-' if negative, and +1 NULL ending character).

;

; Arguments: AX - the signed 16 bit value.

; SI - the location to write the string.

;

; Return Value: Resulting hexadecimal representation ASCII string is written

; at SI.

;

; Local Variables: Argument or Digit (AX)

; Temporary location for remainder to be next argument (BX)

; Remainder (DX)

; pwr10 (CX)

; nextChar (SI)

;

; Shared Variables: None.

; Global Variables: None.

;

; Input: None.

; Output: None.

;

; Error Handling: None.

;

; Algorithms: Repeatedly divide by powers of 10 and get the remainders

; which are the digits.

;

; Data Structures: None.

;

; Registers Used: AX, BX, CX, DX, SI

;

; Stack Depth: 8 words (pushed all general purpose registers)

;

; Author: Archan Luhar

; Last Modified: 10/26/2013

;

;

; Pseudo Code (given n = argument, a = )

; -----------

; If n = 0:

; \*a = '0'

; a++

; else:

; if n < 0:

; \*a = '-'

; a++

; n = -n

; foundFirstNonZero = false

; pwr10 = 10000 (since maximum of 5 digits in 16 bit value)

; while pwr10 > 0:

; int digit = n / pwr10

; n = n mod pwr10

; pwr10 = pwr10 / 10

; if digit != 0 || foundFirstNonZero:

; foundFirstNonZero = true

; \*a = '0' + digit

; a++

; \*a = ASCII\_NULL = 0

; return

Dec2String PROC NEAR

PUBLIC Dec2String

InitDec2String:

PUSHA ; Save all general purpose registers

CheckZero:

CMP AX, 0 ; If argument number is zero, skip

JZ WriteZero ; following and write zero.

CheckNegative: ; If number is negative, write ASCII dash.

JNL SetupPower10

MOV BYTE PTR [SI], '-'

INC SI

NEG AX ; Make the number positive.

SetupPower10: ; pwr10 = 10000

MOV CX, 10000

FindFirstPwr10Loop:

MOV DX, 0 ; Clear DX for 16 bit division

DIV CX ; digit = AX = n / pwr10.

; remainder = DX = n mod pwr10

CMP AX, 0 ; If digit is not zero, nonzero digit found!

JNZ WriteDigit ; If found, go write this and all next.

MOV BX, DX ; Save next argument n mod pwr 10 in BX

MOV AX, CX ; Setup AX for dividing pwr10 by 10

MOV CX, 10

MOV DX, 0

DIV CX

MOV CX, AX ; Move pwr10 / 10 back to CX

MOV AX, BX ; move next (n mod pwr10) back to AX

JMP FindFirstPwr10Loop

WriteDigitsLoop:

MOV DX, 0

DIV CX ; Divide number by power of ten

WriteDigit:

ADD AX, '0' ; Make digit into ASCII character

MOV BYTE PTR [SI], AL ; Write the character

INC SI ; Increment the string pointer to next byte

MoveToNextPwr:

MOV BX, DX ; Save arg mod pwr10 from being overwritten by

; next instructions that produce the next power

MOV AX, CX ; Setup pwr10 for divison

MOV CX, 10

MOV DX, 0

DIV CX

CMP AX, 0

JZ EndDec2String

MOV CX, AX ; pwr10 = pwr10 / 10

MOV AX, BX ; Restore arg mod pwr10 to AX

EndWriteDigitsLoop:

JMP WriteDigitsLoop

WriteZero:

MOV BYTE PTR [SI], '0' ; Skipped digit writing code to write zero

INC SI ; and continue to end the string.

EndDec2String: ; End string with ASCII NULL

MOV BYTE PTR [SI], ASCII\_NULL

POPA ; Restore general purpose registers and return

RET

Dec2String ENDP

; Hex2String

;

; Description: This function is used to create a hexadecimal ascii string

; given an unsigned binary value.

;

; Operation: Given a 16 bit unsigned value, hexadecimal ascii characters

; are written to a specified memory location by going through

; each 4 bit quarter of the binary representation via a

; incrementally shifting and masking.

;

; Arguments: AX - the unsigned 16 bit value.

; SI - the location to write the string.

;

; Return Value: Resulting hexadecimal representation ASCII string is written

; at SI.

;

; Local Variables: original value (AX)

; digit (BX)

; numRightZeroBits (CL)

; mask (DX)

; strPointer (SI)

;

; Shared Variables: None.

; Global Variables: None.

;

; Input: None.

; Output: None.

;

; Error Handling: None.

;

; Algorithms: None.

; Data Structures: None.

;

; Registers Used: AX, BX, CX, DX, SI

; Stack Depth: 8 words for saving all registers

;

; Author: Archan Luhar

; Last Modified: 10/26/2013

;

; Pseudo Code

; -----------

; If n = 0:

; \*a = '0'

; a++

; else:

; foundFirstNonZero = false

; mask = 1111 0000 0000 0000 b

; numRightZeroBits = 12

; while mask > 0:

; digit = n & mask

; digit Right Shift numRightZeroBits

; numRightZeroBits -= 4

; mask Right Shift 4

; if digit != 0 || foundFirstNonZero:

; foundFirstNonZero = true

; if digit < 10:

; \*a = '0' + digit

; else:

; \*a = 'A' + digit

; a++

; \*a = ASCII\_NULL = 0

; return

Hex2String PROC NEAR

PUBLIC Hex2String

InitHex2String:

PUSHA ; Save all general purpose registers

HexCheckZero:

CMP AX, 0 ; If input number is 0, skip everything

JZ WriteHexZero ; And write zero. Else continue.

SetupMask:

MOV DX, 0f000h ; Sets up mask to get highest four bits

MOV CL, 12 ; Sets up the number of 0 bits to the right

FindFirstNonZeroLoop: ; Loop until the first non zero digit is

TEST AX, DX ; found.

JNZ WriteHexDigitsLoop ; If so, move on to writing it.

SHR DX, 4 ; If not, shift the mask

ADD CL, -4 ; and correct the number of bits to right.

JMP FindFirstNonZeroLoop

WriteHexDigitsLoop:

MOV BX, AX ; Setup BX to store the digit

AND BX, DX ; Get the digit by AND'ing the mask

SHR BX, CL ; Make sure digit has no trailing zeroes

SHR DX, 4 ; Update the mask for the next digit.

ADD CL, -4 ; And update the number of bits to the right

DecideOffset:

CMP BX, 10 ; If the digit is less than 10

JL AddDigitOffset ; add the ASCII digit offset.

AddLetterOffset: ; Else, add the ASCII letter offset.

ADD BX, -10 ; (A 10 corresponds to an 'A')

ADD BX, 'A'

JMP WriteHexDigit ; And write it.

AddDigitOffset:

ADD BX, '0' ; Add the digit offset. Continue to write.

WriteHexDigit:

MOV BYTE PTR [SI], BL ; Write the character created from offseting

INC SI ; Increment the string pointer to next byte

EndWriteHexDigitsLoop:

CMP DX, 0 ; If mask is zero, no more digits left,

JZ EndHex2String ; finish off the string and function.

JMP WriteHexDigitsLoop ; Else, write next digit.

WriteHexZero:

MOV BYTE PTR [SI], '0' ; Skipped digit writing code to write zero

INC SI ; and continue to end the string.

EndHex2String:

MOV BYTE PTR [SI], ASCII\_NULL

POPA ; Restore all general purpose registers

RET

Hex2String ENDP

CODE ENDS

END